Аналитика данных с помощью pandas и matplotlib

В этом задании вам предлагается выполнить анализ данных криптовалют с помощью библиотек pandas и matplotlib. Задание выглядит как лабораторная работа, в которой вам предстоит заполнить недостающие клетки и ответить на ряд вопросов.

Минимальные баллы для зачёта по этой работе - 3 балла. Если вы не набираете тут 3 балла, то по всему курсу вы получаете неуд (см. слайды с семинара №1)

- Официальная документация pandas (https://pandas.pydata.org/)
- Официальная документация по matplotlib (https://matplotlib.org/index.html)

1. Данные (суммарно 2 балла)

Начнем с необходимых приготовлений.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import ipywidgets  # Библиотека для интерактивных контролов в jupyter notebook'е

%matplotlib inline
```

Загрузите заранее подготовленный датасет из файла "coins.csv". Создайте объект типа pandas.DataFrame с именем coins и в качестве индекса выберите колонку с датой.

In [2]:
```
coins = pd.read_csv('coins.csv', index_col='date')
```

Посмотрим что получилось

In [3]:
```
coins.head(4)
```

Out[3]:

date	price	txCount	txVolume	activeAddresses	symbol	name	open	high	low	close	volume	market
2013-04-28	135.30	41702.0	6.879868e+07	117984.0	BTC	Bitcoin	135.30	135.98	132.10	134.21	0.0	1.500520e+09
2013-04-28	4.30	9174.0	4.431952e+07	17216.0	LTC	Litecoin	4.30	4.40	4.18	4.35	0.0	7.377340e+07
2013-04-29	134.44	51602.0	1.138128e+08	86925.0	BTC	Bitcoin	134.44	147.49	134.00	144.54	0.0	1.491160e+09
2013-04-29	4.37	9275.0	3.647810e+07	18395.0	LTC	Litecoin	4.37	4.57	4.23	4.38	0.0	7.495270e+07

Поясним значения хранящиеся в колонках

- date - дата измерений
- name - полное название монеты
- symbol - сокращенное название монеты
- price - средняя цена монеты за торговый день в USD
- txCount - количество транзакций в сети данной монеты
- txVolume - объем монет переведенных между адресами в сети данной монеты
- activeAddresses - количество адресов совершавших а данный день транзации в сети данной монеты
- open - цена монеты в начале торгов данного дня
- close - цена монеты в конце торгов данного дня
- high - самая высокая цена данной монеты в течение данного торгового дня
- low - самая низкая цена данной монеты в течение данного торгового дня
- volume - объем торгов данной монетой на биржах в данный день
- market - капитализация данной монеты в данный день

Изучим полученные данные. Ответьте на следующие вопросы (вставляйте клетки с кодом и тектом ниже):

1. Сколько всего различных монет представлено в датасете? (0.4 балла)

In [4]:
```
len(np.unique(coins['symbol'].values))
```

Out[4]:
```
66
```

2. За какой период данные мы имеем? (0.4 балла)

In [5]:
```
print('from', coins.index.min(), 'to', coins.index.max())
```

```
from 2013-04-28 to 2018-06-06
```

3. Есть ли пропуски в данных? Какой природы эти пропуски? (0.5 балла)

In [7]:
```
for attr in coins.columns:
    print(attr, " - ", coins[attr].isnull().values.sum())
```

```
price  -  327
txCount  -  1520
txVolume  -  1830
activeAddresses  -  1520
symbol  -  0
name  -  0
open  -  0
high  -  0
low  -  0
close  -  0
volume  -  0
market  -  0
```

4. У какой монеты и когда была самая высокая цена? (0.2 балла)

In [6]:
```
display(coins[coins['high'] == coins['high'].max()][['symbol', 'high']])
```

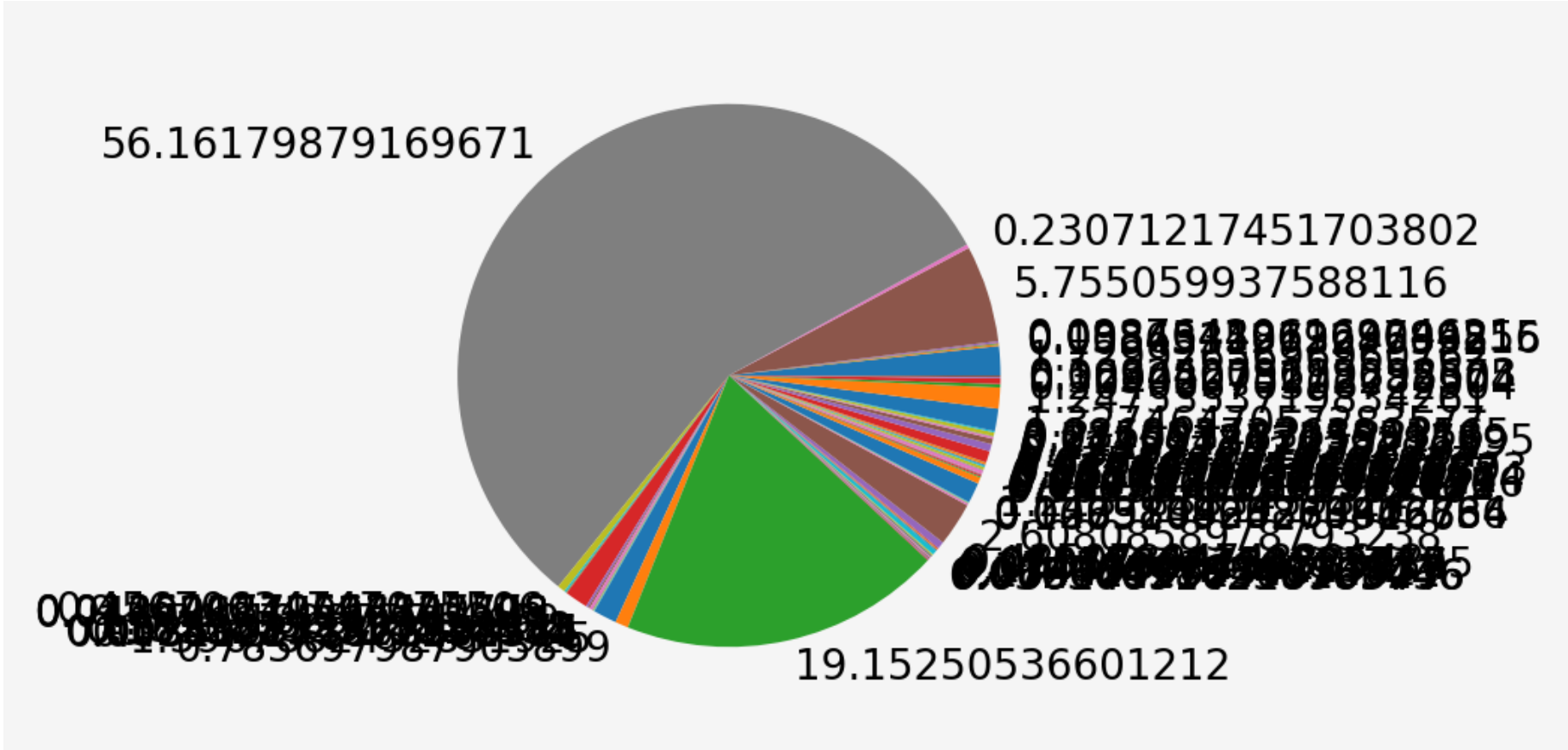
date	symbol	high
2017-12-17	BTC	20089.0

5. У какой монеты самая высокая и самая низкая суммарная капитализация? Постройте круговую диаграмму с долями. (0.5 балла)

In [13]:
```
sum_market = coins.groupby(['symbol'])['symbol','market'].sum()


display(sum_market[sum_market['market'] == sum_market['market'].max()])
display(sum_market[sum_market['market'] == sum_market['market'].min()])


all_sum = sum_market['market'].sum()

fig = plt.figure(
    figsize=(3, 3),
    facecolor='whitesmoke',
    dpi=200
)

plt.pie(
    sum_market['market'].values,
    labels = list(map(lambda x: (x*100/all_sum), sum_market['market'].values),)
)

plt.show()
```

symbol	market
BTC	5.743947e+13

symbol	market
CTXC	1.093502e+10


56.16179879169671
0.23071217451703802
5.755059937588116
19.15250536601212

2. Визуализация (1 балл)

Самая интересная часть работы аналитика состоит во внимательном вглядывании в правильно выбранные и построенные графики.

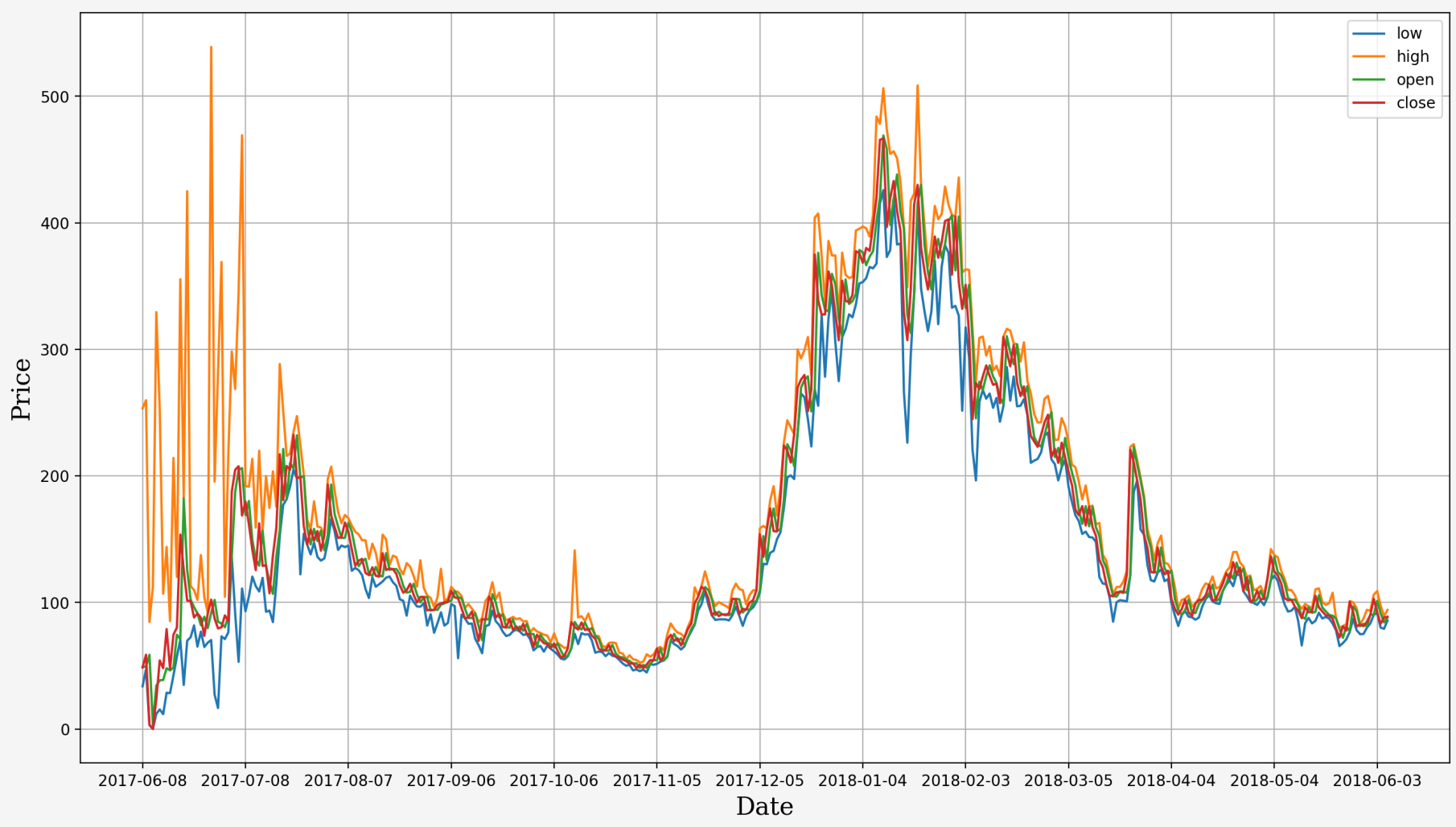
Реализуйте функцию для визуализации цен выбранной валюты за выбранный диапазон дат.

На графике должны быть видны цены начала и конца продаж. А так же минимальная и максимальная цена за этот день. Подпишите график и оси координат. Добавьте сетку. Увеличьте размер изображения. Можете попробовать использовать matplotlib.finance.candlestick_ohlc (mpl_finance.candlestick_ohlc), но можно и без него.

In [15]:
```
def plot_fancy_price_action(coins, symbol, start_date, end_date):
    main = coins[coins['symbol'] == symbol][['low', 'high', 'open', 'close']]
    price_date = main[start_date:end_date][['low', 'high', 'open', 'close']]
    fig = plt.figure(
        figsize=(16, 9),
        facecolor='whitesmoke',
        dpi=200
    )

    plt.plot(
        price_date.index,
        price_date['low'].values,
        label='low'
    )
    plt.plot(
        price_date['high'].values,
        label='high'
    )

    plt.plot(
        price_date['open'].values,
        label='open'
    )
    plt.plot(
        price_date['close'].values,
        label='close'
    )
    plt.xlabel(
        'Date',  # Текст
        fontdict=dict(family='serif', color='black', weight='normal', size=16)\
    )
    plt.ylabel(
        'Price',
        fontdict=dict(family='serif', color='black', weight='normal', size=16)
    )
    plt.xticks(range(0, len(price_date.index), 30), )
    plt.legend()
    plt.grid(True)
    plt.show()
```

Посмотрим, что получилось:

In [16]:
```
plot_fancy_price_action(coins=coins, symbol='VERI', start_date='2013-06-01', end_date='2019-06-30')
```



Никакого датасаенса в этом задании нет. Просто аналитик должен уметь строить графики, либо знать готовые инструменты.

3. Накачка и сброс (1 балл)

Криптовалютные биржи до сих пор остаются маргинальным местом, эдаким диким западом финансового мира. Как следствие, здесь процветают схемы относительно честного отъема денег. Одна из них - pump'n'dump (накачка и сброс). Она выглядит следуюшим образом. Несколько крупных игроков или много мелких договариваются вместе купить малоизвестную монету с низкой ценой и объемом торгов. Это приводит к мгновенному взлету цены (pump), далее приходят неопытные игроки в надежде успеть заработать на таком росте. В этот момент организаторы схемы начнают все продавать (dump). Весь процесс занимает от нескольких минут до нескольких часов.

Ваша задача найти самый сильный pump'n'dump монеты на заданном промежутке времени. Для этого для каждого дня определим число pnd равное отношению максимальной цены монеты в данный день к максимуму из цен открытия и закрытия в тот же день. Нужно найти день когда pnd был максимален и величину pnd.

In [19]:
```
def find_most_severe_pump_and_dump(coins, symbol, start_date, end_date):
    tmp = coins[coins['symbol'] == symbol][[ 'high', 'open', 'close']]
    pnd = coins[start_date:end_date][[ 'high', 'open', 'close']]
    pnd['max'] = [max(a, b) for a, b in zip(pnd['open'].values, pnd['close'].values)]
    pnd['pnd'] = [a/b for a, b in zip(pnd['high'].values, pnd['max'].values)]
    display(pnd[pnd['pnd'] == pnd['pnd'].max()][['pnd']])
```

In [20]:
```
find_most_severe_pump_and_dump(coins, symbol='BTC', start_date='2017-06-01', end_date='2018-06-01')
```

	pnd
date	
2017-06-11	33.549254

Сравните эти значения для разных монет.

In [6]:
```
sd='2017-06-01'
ed='2018-06-01'
pnd = coins[sd:ed][['symbol', 'high', 'open', 'close']]
pnd['max'] = [max(a, b) for a, b in zip(pnd['open'].values, pnd['close'].values)]
pnd['pnd'] = [a/b for a, b in zip(pnd['high'].values, pnd['max'].values)]
pnd.groupby(['symbol'])['symbol','pnd'].max().sort_values(by=['pnd'])
```

Out[6]:

	symbol	pnd
symbol		
USDT	USDT	1.099010
LTC	LTC	1.108226
XEM	XEM	1.135870
CTXC	CTXC	1.142857
BTC	BTC	1.142894
ETH	ETH	1.143351
ETC	ETC	1.148249
NAS	NAS	1.175824
BNB	BNB	1.176080
CENNZ	CENNZ	1.177753
DASH	DASH	1.203162
DRGN	DRGN	1.211765
SALT	SALT	1.227207
NEO	NEO	1.232143
BAT	BAT	1.234249
ICX	ICX	1.253846
WAVES	WAVES	1.254335
ZRX	ZRX	1.254990
VEN	VEN	1.271357

	symbol	pnd
symbol		
DOGE	DOGE	1.274733
ELF	ELF	1.280899
ADA	ADA	1.290640
MAID	MAID	1.291362
EOS	EOS	1.320293
KCS	KCS	1.324376
ZEC	ZEC	1.329878
OMG	OMG	1.331215
XLM	XLM	1.332511
PAY	PAY	1.351402
PIVX	PIVX	1.358796
...	...	...
VTC	VTC	1.395210
LSK	LSK	1.442478
POLY	POLY	1.455377
ZIL	ZIL	1.463186
GAS	GAS	1.477106
REP	REP	1.477601
WTC	WTC	1.498617
LOOM	LOOM	1.526513
GNO	GNO	1.653660
QASH	QASH	1.665838
MTL	MTL	1.672854
GNT	GNT	1.688212
CVC	CVC	1.726167

symbol	symbol	pnd
XVG	XVG	1.726865
ANT	ANT	1.733668
AE	AE	1.748294
BCH	BCH	1.783945
DCR	DCR	1.805444
KNC	KNC	1.851924
SRN	SRN	1.891649
ICN	ICN	2.586337
MANA	MANA	2.970278
PPT	PPT	3.720798
LRC	LRC	4.528409
RHOC	RHOC	5.075208
BTG	BTG	5.777033
BTM	BTM	7.177933
TRX	TRX	9.651010
FUN	FUN	12.490562
VERI	VERI	33.549254

66 rows × 2 columns

4. Окупаемость инвестиций (1 балл)

Вам нужно посчитать окупаемость инвестиций в криптовалюты на заданном промежутке времени. Окупаемость определяется как отношение изменения цены портфеля к исходной цене портфеля. Цена портфеля - это суммарная стоимость (в USD) всех

монет в портфеле.

investments - dict в котором ключи - это названия монет, значения - это сумма вложений в эту монету (в USD)

In [24]:
```
def compute_roi(coins, investments, start_date, end_date):
    result = {}
    for key in investments.keys():
        tmp = coins[start_date:start_date]
        open_price = tmp[tmp['symbol'] == key]['open'].values[0]
        tmp = coins[end_date:end_date]
        close_price = tmp[tmp['symbol'] == key]['close'].values[0]
        result[key] = {'input': investments[key], 'output':  investments[key]*close_price/open_price,'start'
    delta = 0
    sum_input = 0;
    for key in result.keys():
        sum_input += result[key]['input']
        delta += (result[key]['output'] - result[key]['input'])
    return delta/sum_input
```

In [25]:
```
compute_roi(coins, investments={'BTC': 1000, 'LTC': 500}, start_date='2018-04-04', end_date='2018-06-01')
```

Out[25]:
```
-0.02722108303745222
```

In [26]:
```
compute_roi(coins, investments={'BTC': 1000, 'LTC': 500}, start_date='2013-05-28', end_date='2018-06-06')
```

Out[26]:
```
51.35085448945653
```

5. Технический анализ (1 балл)

Технический анализ это способ предсказания поведения графика по некоторым вспомогательным величинам построенным по исходному графику. Один из простейших методов технического анализа - границы Болинджера. Кто-то верит, что график касаясь границы от него должен отражаться.

Нарисуйте график цены, скользящее среднее и границы Боллинджера (https://en.wikipedia.org/wiki/Bollinger_Bands) с параметрами N (window) = 21, K (width) = 2.

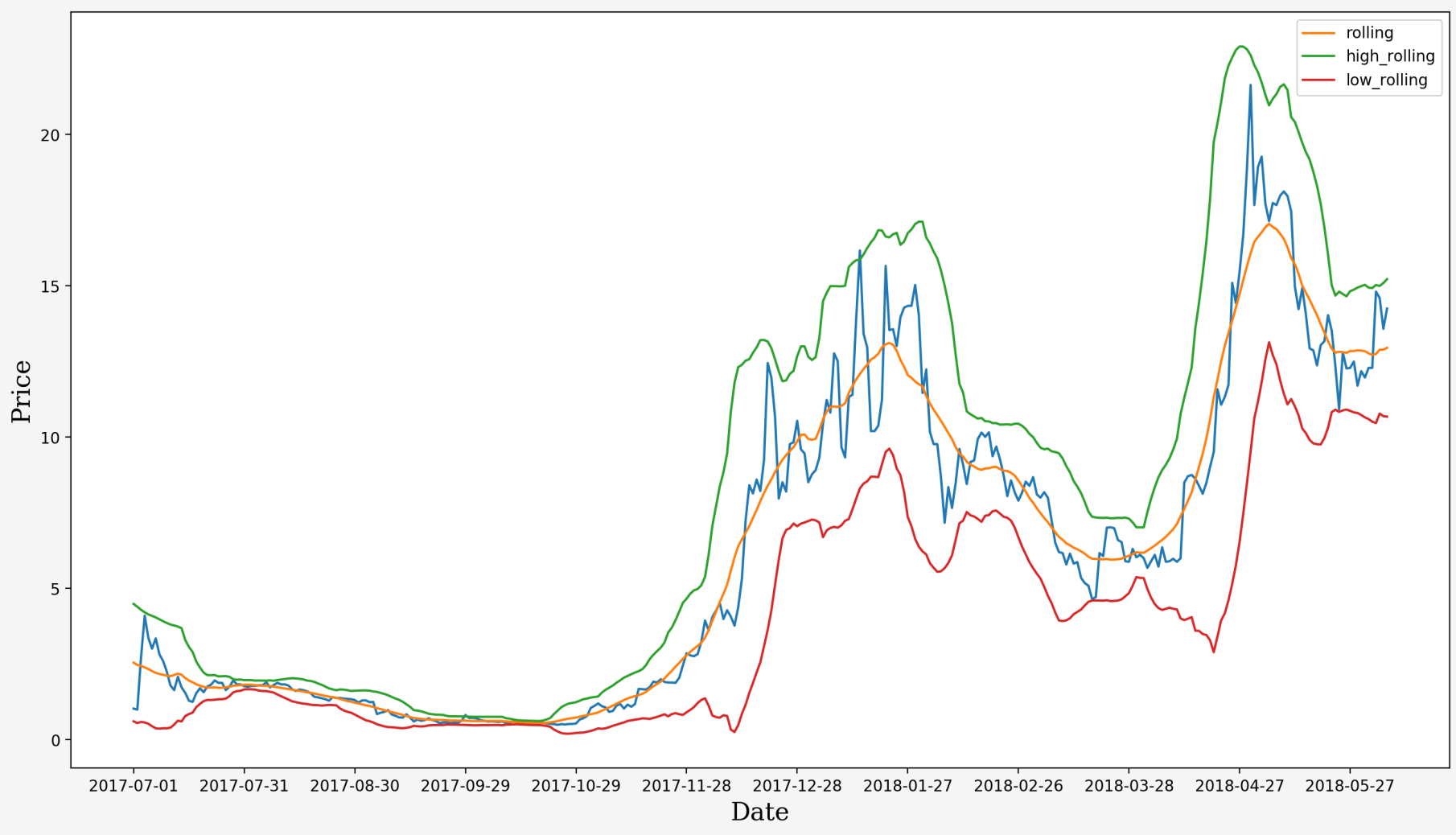
Границы считаются очень просто: (MA + Kσ) и (MA - Kσ), где MA - скользящее среднее за N дней, а σ - скользящее стандартное отклонение за N дней.

Тут вам поможет функция rolling для подсчёта среднего и стандартного отклонения по скользящему окну.

Не забудьте подписать график и оси, отрисовать легенду и выбрать для нее лучшее расположение.

In [28]:
```
def plot_bollinger_bands(coins, symbol, window, width):
    fig = plt.figure(
        figsize=(16, 9),
        facecolor='whitesmoke',
        dpi=200
    )
    price = coins[coins['symbol'] == symbol]['price']
    leading_mean = price.rolling(window=window, center=True, min_periods=1).mean()
    leading_std = price.rolling(window=window, center=True,  min_periods=1).std()


    plt.plot(coins[coins['symbol'] == symbol]['price'].values)
    plt.plot(leading_mean, label='rolling')
    plt.plot(leading_mean + width * leading_std, label='high_rolling')
    plt.plot(leading_mean - width * leading_std, label='low_rolling')
    plt.xticks(range(0, len(price.index), 30))
    plt.xlabel(
        'Date',  # Текст
        fontdict=dict(family='serif', color='black', weight='normal', size=16)\
    )
    plt.ylabel(
        'Price',  # Текст
        fontdict=dict(family='serif', color='black', weight='normal', size=16)\
    )
    plt.legend()
    plt.show()
```

In [29]:
```
plot_bollinger_bands(coins=coins, symbol='EOS', window=21, width=2)  # тут должен появиться график
```



Сделайте вывод о том, выполнялось ли правило Боллинджера.

Type *Markdown* and LaTeX: α^2

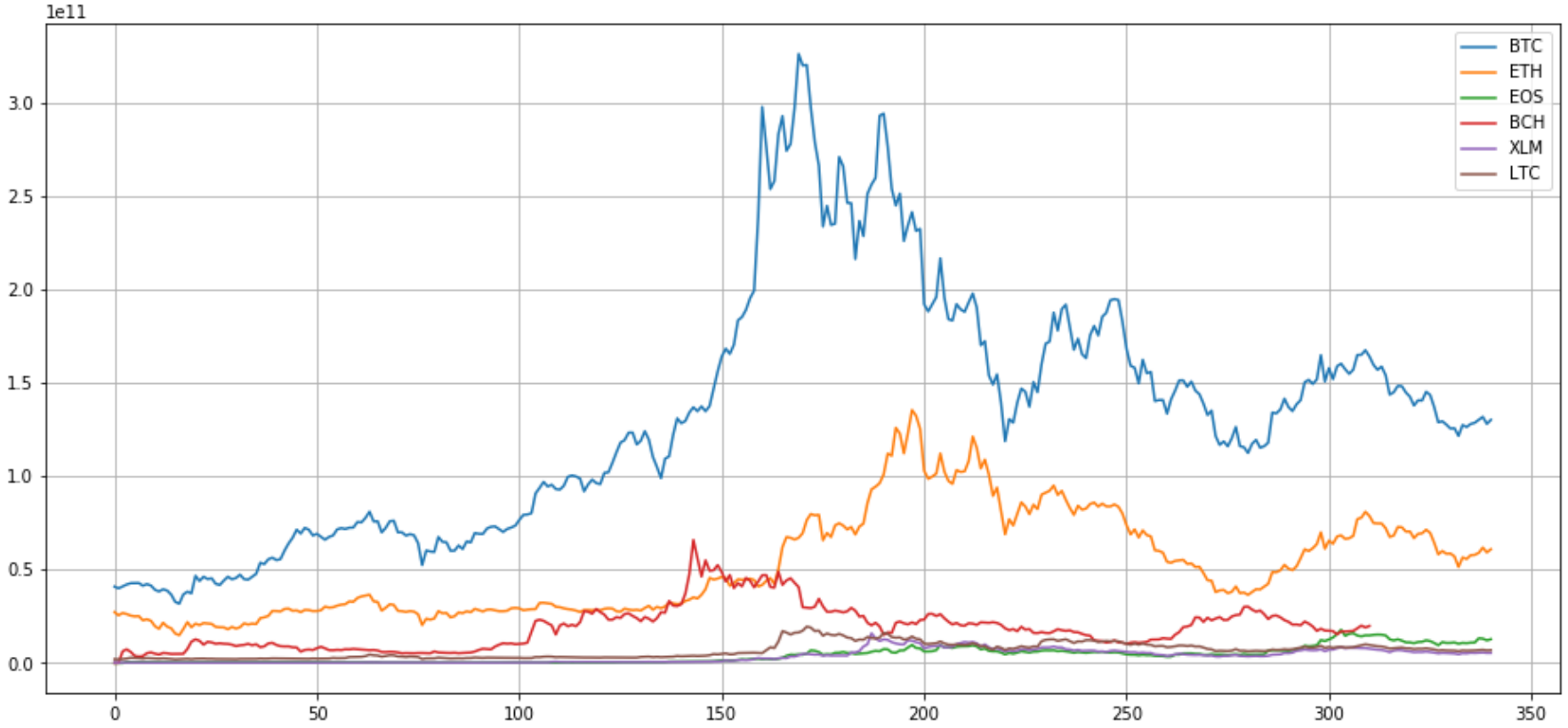
6. Капитализация как индикатор (1 балл)

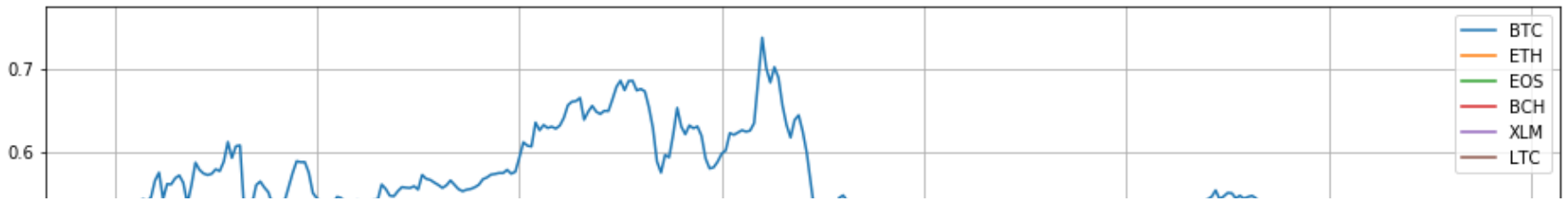
Многие люди, которые торгуют криптовалютой, любят смотреть на капитализацию. Давайте поймём почему.

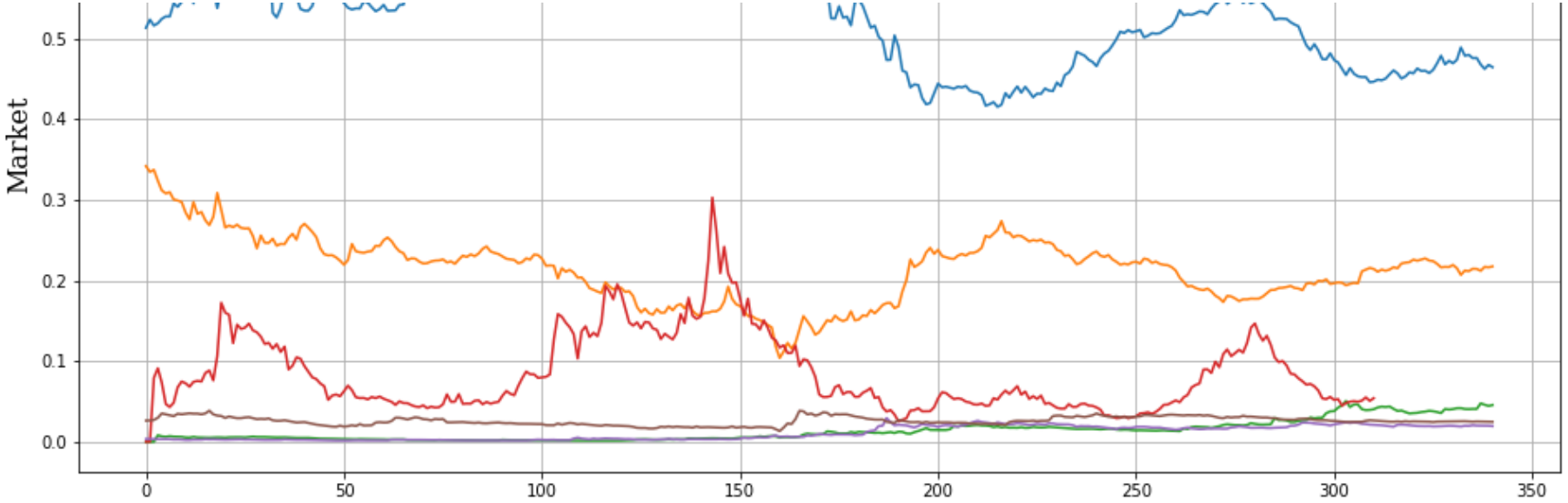
Нарисуйте еще два графика. На первом должна быть общая капитализация биткойна (BTC), эфира (ETH), eос (EOS), биткойн кэша (BCH), стеллара (XLM) и лайткойна (LTC). На втором - доли капитализаций этих монет от общей капитализации рынка. При этом используйте данные начиная с 2017-07-01.

In [32]:
```
def plot_coins_capitalizations(coins, symbols, start_date):
    symb_market = {}

    fig, (ax1, ax2) = plt.subplots(
    nrows=2, ncols=1,
    figsize=(16, 16)
    )
    tmp = coins[start_date:]
    for symb in symbols:
        symb_market[symb] = tmp[tmp['symbol'] == symb]['market'].values
    for symb in symb_market.keys():
        ax1.plot(symb_market[symb], label=symb)
    ax1.legend()
    ax1.grid(True)

    sum_market = tmp.groupby('date')['market'].sum().values
    symb_share = {}
    for symb in symb_market.keys():
        symb_share[symb] = [a / b for a, b in zip(symb_market[symb], sum_market)]
    for symb in symb_share.keys():
        ax2.plot(symb_share[symb], label=symb)
    ax2.legend()
    ax2.grid(True)
```

In [33]:
```
plot_coins_capitalizations(
    coins=coins,
    symbols=('BTC', 'ETH', 'EOS', 'BCH', 'XLM', 'LTC'),
    start_date='2017-07-01'
)
```







Проанализируйте зависимость доли капитализации альткойнов от доли капитализации биткойна. Как выдумаете, в чём причина такой зависимости?

без понятия что от меня тут хотят, но вот...

Если смотреть на зависимость BCH и BTC, то наблюдается схожее поведение графиков(периоды возрастания и убывания совпадают). С ETH и BTC ситуация противоположная, те периоды возрастания и убывания у них обратны.

Возмоно такая зависимость связанная с тем, что из себя представляют альткойны, тк я не стал разбираться в этом

Type *Markdown* and LaTeX: α^2

7. Корреляции монет (1 балл)

Теперь нужно подробнее посмотреть на корреляции средних капитализаций монет. При этом будем смотреть на среднее сглаженное за последние window дней до дня date с коэффициентом сглаживания alpha для набора монет symbols.

Реализуйте функцию, которая будет возвращать квадратный DataFrame с числом строк и столбцов равным числу рассматриваемых монет и со значениями корреляций.

In [28]:
```
def calc_coins_correlations(coins, date, symbols, window, alpha):
    # Paste your code here
```

In [ ]:
```
correlations = calc_coins_correlations(coins, date="2018-06-06",
                                       symbols=['BTC', 'ETH', 'EOS', 'BCH', 'XLM', 'LTC', 'ADA'],
                                       window=21, alpha=0.1)
# Теперь посмотрим на эти корреляции следующим образом:
correlations.style.background_gradient(cmap='coolwarm').set_precision(2)
```

Довольно интересно ещё взглянуть на 2017-12-27:

In [ ]:
```
correlations = calc_coins_correlations(coins, date="2017-12-27",
                                       symbols=['BTC', 'ETH', 'EOS', 'BCH', 'XLM', 'LTC', 'ADA'],
                                       window=21, alpha=0.1)
# Теперь посмотрим на эти корреляции следующим образом:
correlations.style.background_gradient(cmap='coolwarm').set_precision(2)
```

8. Анализ одной стратегии (2 балла)

Разберем один мечтательный пример. Посмотрим какую прибыль могла бы нам принести хрестоматийная торговая стратегия основанная на скользящих средних. Стратегия выглядит следующим образом: мы строим две скользящие среднии для графика цены. С маленьким окном (ведущее скользящее среднее) и с бОльшим окном (запаздывающее скользящее среднее). Мы покупаем, когда ведущее среднее становится больше запаздывающего, и продаем в противном случае. Посмотрим на пример

In [96]:
```
def plot_moving_averages(coins, symbol, leading_window, lagging_window, start_date, end_date):
    coin = coins[coins['symbol'] == symbol][start_date:end_date]
    price = coin['price']
    leading_mean = price.rolling(window=leading_window).mean()
    lagging_mean = price.rolling(window=lagging_window).mean()

    fig = plt.figure(figsize=(16, 9))
    ax = fig.add_subplot(111)

    ax.set_title('Price action for {}'.format(symbol))
    ax.plot(leading_mean, color='green', label='MA{}'.format(leading_window))
    ax.plot(lagging_mean, color='red', label='MA{}'.format(lagging_window))
    ax.plot(price, color='blue', label='price')
    ax.set_xlabel('Date')
    ax.set_ylabel('Price')
    ax.legend(loc='best')
    ax.grid(True)
    plt.show()

plot_moving_averages(
    coins=coins,
    symbol='BTC',
    leading_window=21,
    lagging_window=50,
    start_date='2017-05-01',
    end_date='2018-08-01')
```

Видно, что для скользящее среднее с бОльшим окном медленнее реагирует на изменение цены. Именно на этой идее и основана торговая стратегия.

Реализуйте функцию, которая строит два графика. На правом будут изображены цена и скользящие средние. На левом - во сколько раз изменится размер вложений при использовании нашей стратегии и при обычном инвестировании

Notes:

Давайте использовать только цены закрытия. При этом, чтобы узнать цены за вчерашний день, стоит использовать метод shift(1) у Series. Отношение цен закрытия за сегодня и за вчера - это мой multiplier за сегодняшний день. При этом давайте строить графики накопления для multipliers. Т.е. если мы смотрим на 3 дня и в первый день multiplier = 1.5, во второй- 0.5 и в третий 2. То график будет выглядеть так: (1.5, 1.5 * 0.5, 1.5 * 0.5 * 2).

При использовании нашей новой стратегии мы будем либо покупать, если ведущее среднее становится больше запаздующего на некоторый threshold (при этом лучше разницу сперва поделить на цену), либо оставлять всё как есть. При этом, конечно, нужно, принимая решения за сегодняшний день, смотреть только на статистику из прошлого.

In [32]:
```
def plot_moving_averages_strategy(
    coins, symbol, lead_window, lag_window, threshold, start_date, end_date
):
    # Paste your code here
```

In [ ]:
```
# Теперь на основе реализованной функции сделаем интерактивные графики и поизучаем, что получилось:
symbol_selector = ipywidgets.Dropdown(
    options=('BTC', 'ETH', 'EOS', 'BCH', 'XLM', 'LTC', 'ADA'),
    index=0,
    value='BTC',
    layout={'width': '700px'},
    continuous_update=False
)

lead_window_slider = ipywidgets.IntSlider(
    value=21,
    min=1,
    max=200,
    step=1,
    layout={'width': '700px'},
    continuous_update=False)

lag_window_slider = ipywidgets.IntSlider(
    value=50,
    min=1,
    max=200,
    layout={'width': '700px'},
    step=1, continuous_update=False)

threshold_slider = ipywidgets.FloatSlider(
    min=0,
    max=0.20,
    step=0.001,
    value=0.025,
    layout={'width': '700px'},
    continuous_update=False)

start_date_slider = ipywidgets.SelectionSlider(
    options=pd.date_range('2013-04-28', '2018-06-06', freq='D'),
    index=0,
    value=pd.Timestamp('2017-05-01'),
    layout={'width': '700px'},
    continuous_update=False
)

end_date_slider = ipywidgets.SelectionSlider(
```

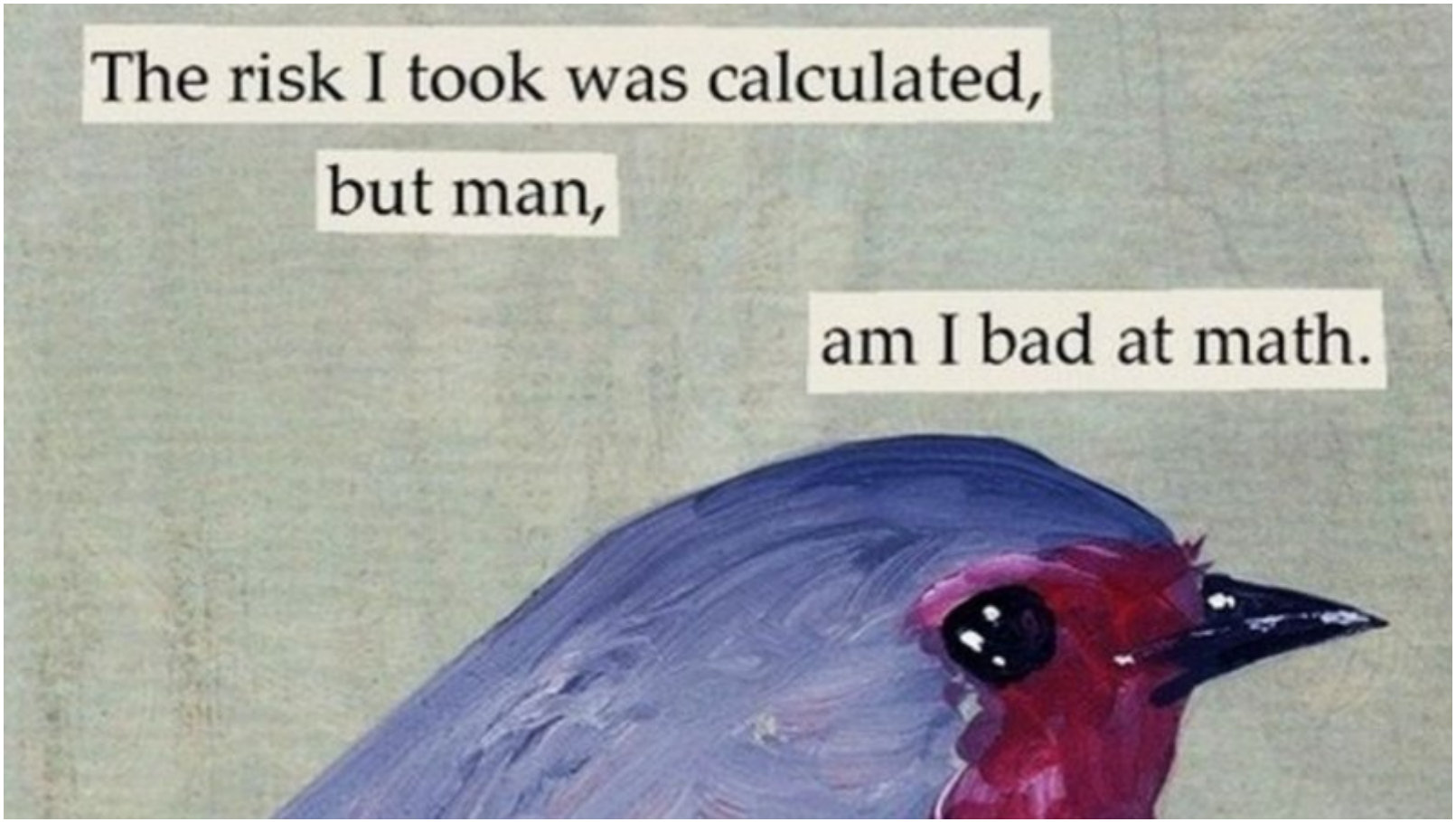
```
    options=pd.date_range('2013-04-28', '2018-06-06', freq='D'),
    index=0,
    value=pd.Timestamp('2018-01-01'),
    layout={'width': '700px'},
    continuous_update=False
)

ipywidgets.interact(
    plot_moving_averages_strategy,
    coins=ipywidgets.fixed(coins),
    symbol=symbol_selector,
    lead_window=lead_window_slider,
    lag_window=lag_window_slider,
    threshold=threshold_slider,
    start_date=start_date_slider,
    end_date=end_date_slider
)
```

Попробуйте разные значения параметров для разных монет и сделайте выводы о применимости такой модели.

Type *Markdown* and LaTeX: α^2

9. Отказ от ответственности

Все примеры разобранных здесь стратегий являются игрушечными и не подходят для реальной торговли на бирже. Без серьезной подготовки вас там съедят с потрохами.


The risk I took was calculated,
but man,
am I bad at math.